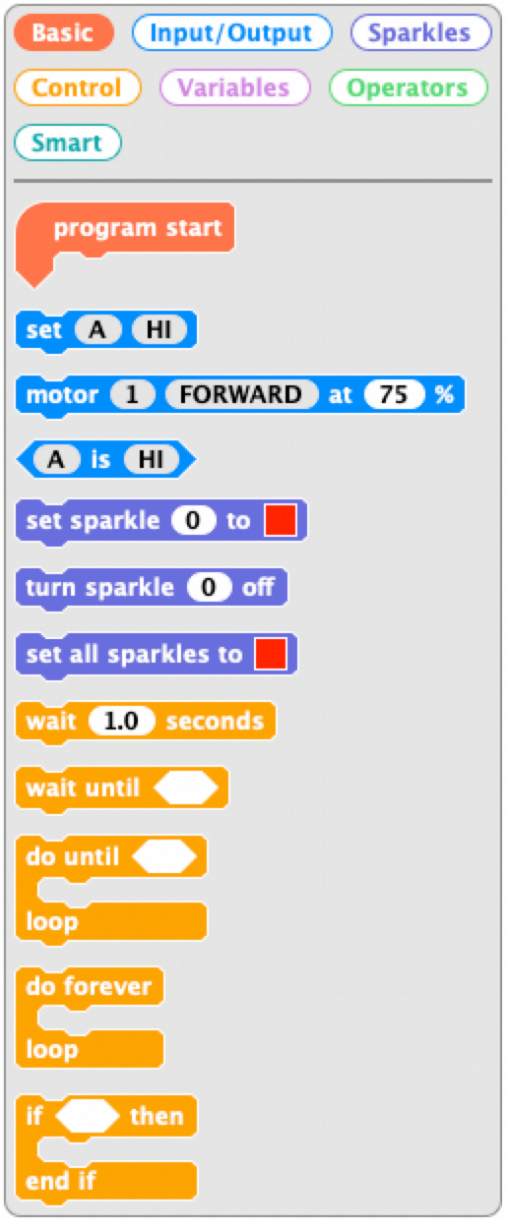


PALETA BASIC

Program start (inicio do programa): *É o primeiro bloque do programa. Só pode haber un bloque deste tipo.*

Set (control de saídas dixitais): *Controla calquera dos terminais (A, B, C, D) como unha saída dixital (Estado: HI = High - 5V ou LO = Low - 0V)*

Motor (control de motores): *Controla o estado (FORWARD: avance de 0 A 100%; REVERSE: retroceso de 0 a 100% e STOP: parada) dos motores (1 o 2)*

Comparación dixital: *Comproba o estado dixital ao que se atopa conectado un terminal (A, B, C, D). Retorna un verdadeiro ou falso no caso de coincidir ou non, a comparación que poñemos no bloque co estado do terminal (HI = High = 5V ou LO = Low = 0V)*

Set sparkle (fixa a cor dun Sparkle): *Permite seleccionar a cor da luz que emitirá o led RGB dun Sparkle (un valor de 0 a 31). Ao facer click no cadro de cor podemos seleccionar a luz que emitirá.*

Turn sparkle off (apaga Sparkle): *Apaga o Sparkle seleccionado.*

Set all sparkles to (fixa a cor de todos os sparkles): *Prende todos os Sparkles que estean conectados a un controlador Crumble cunha mesma cor.*

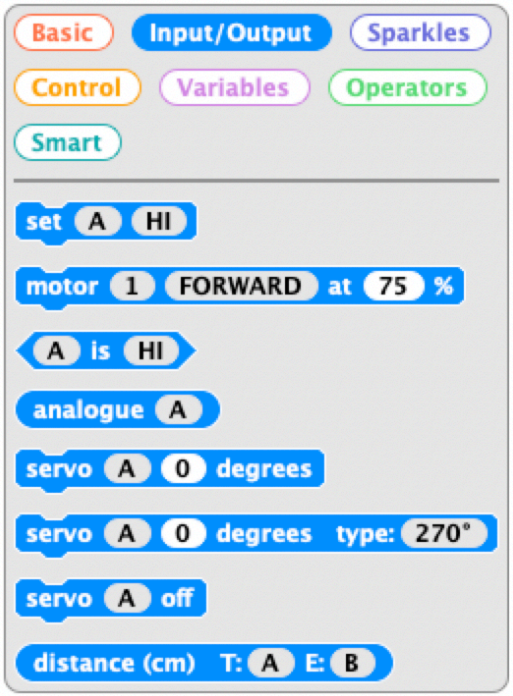
Wait seconds (espera segundos): *Fixa un tempo de espera, en segundos (un valor entre 0 e 32767), entre dous bloques.*

Wait until (espera ata): *Detén a execución do programa ata que se cumpra unha determinada condición lóxica.*

Do until loop (repetir ata): *Executa un conxunto de bloques de programa de forma repetida, ata que se dea a condición que poñamos no rombo da estructura.*

Do forever loop (repetir por sempre): *Executa un conxunto de bloques de programa de forma repetida.*

If - then (facer si): *estrutura condicional. O conjunto de bloques execútanse se se da a condición que poñamos no rombo.*



PALETA INPUT/OUTPUT

Set (control de saídas dixitais): Controla calquera dos terminais (A, B, C, D) como unha saída dixital (Estado: HI = High - 5V ou LO = Low - 0V)

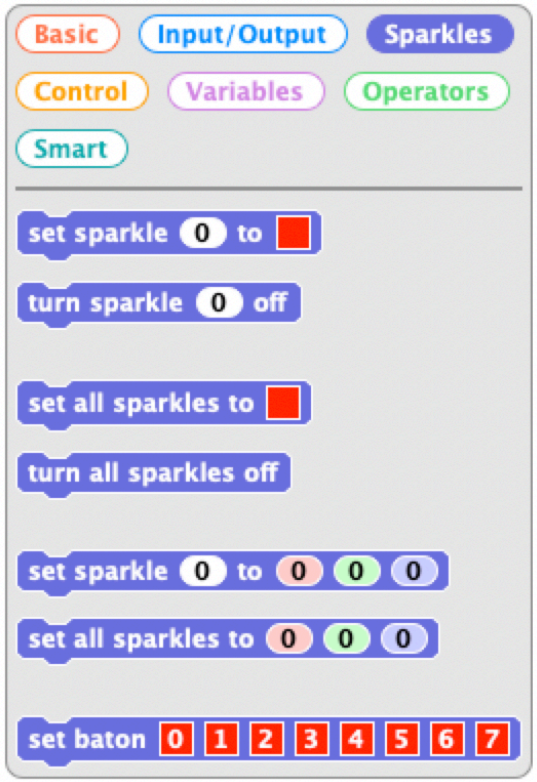
Motor (control de motores): Controla o estado (FORWARD: avance de 0 a 100%; REVERSE: retroceso de 0 a 100% e STOP: parada) dos motores (1 ou 2)

Comparación dixital: Comproba o estado dixital ao que se atopa conectado un terminal (A, B, C, D). Retorna un verdadeiro ou falso no caso de coincidir ou non, a comparación que poñemos no bloque co estado do terminal (HI = High = 5V ou LO = Low = 0V)

Analogue (lectura analóxica dunha entrada): Le o valor analóxico dun sinal conectado a uno dos terminais (A, B, C, D)

Servo (control de servos de RC): Axusta a posición dun servo (-90º a 90º) conectado a un dos terminais (A, B, C, D)

Distance (medida de distancia): Mide a distancia proporcionada por un sensor ultrasónico (Terminal T - *Trigger*: A, B, C, D; Terminal E - *Echo*: A, B, C, D)



PALETA SPARKLES

Set sparkle (fija el color de un Sparkle): Permite seleccionar el color de la luz que emitirá el led RGB de un Sparkle (un valor de 0 a 31). Al hacer click en el cuadro de color podemos seleccionar la luz que emitirá.

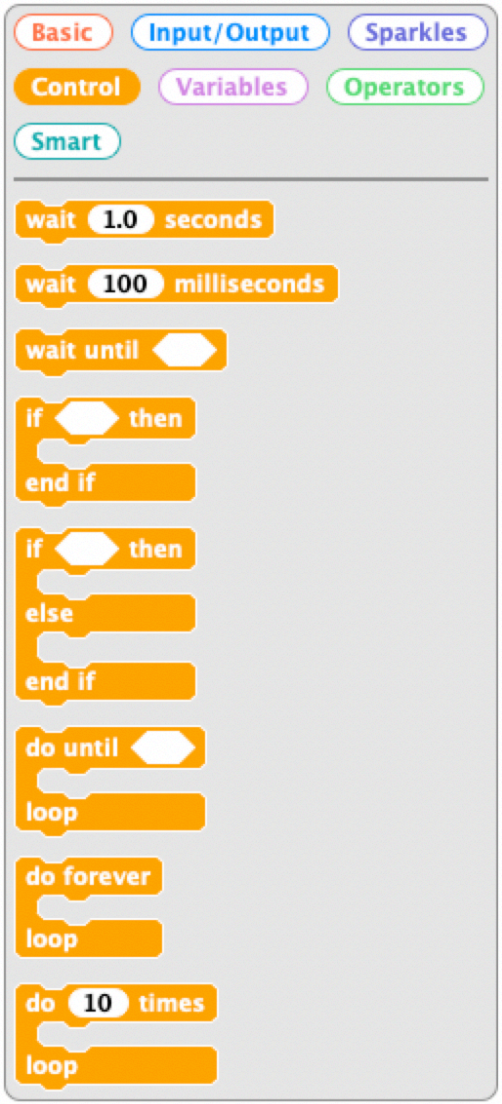
Turn sparkle off (apaga Sparkle): Apaga el Sparkle seleccionado.

Set all sparkles to (fija el color de todos los sparkles): Enciende todos los Sparkles que estén conectados a un controlador Crumble con un mismo color.

Turn all sparkles off (apaga todos los sparkles): Apaga todos los Sparkles.

Set sparkle to RGB (fija el color de un Sparkle con las componentes RGB): Permite seleccionar el color de la luz que emitirá el led RGB de un Sparkle, en función de las componentes RGB (rojo, verde y azul; un valor de 0 a 255)

Set all sparkle to RGB (fija el color de todos los Sparkle con las componentes RGB): Permite seleccionar el color de la luz que emitirá el led RGB de todos los Sparkles, en función de las componentes RGB (rojo, verde y azul; un valor de 0 a 255)



PALETA CONTROL

Wait seconds (espera segundos): Fija un tiempo de espera, en segundos (un valor entre 0 y 32767), entre dos bloques.

Wait milliseconds (espera milisegundos): Fija un tiempo de espera, en milisegundos (un valor entre 0 y 32767), entre dos bloques.

Wait until (espera hasta): Detiene la ejecución del programa hasta que se cumpla una determinada condición lógica.

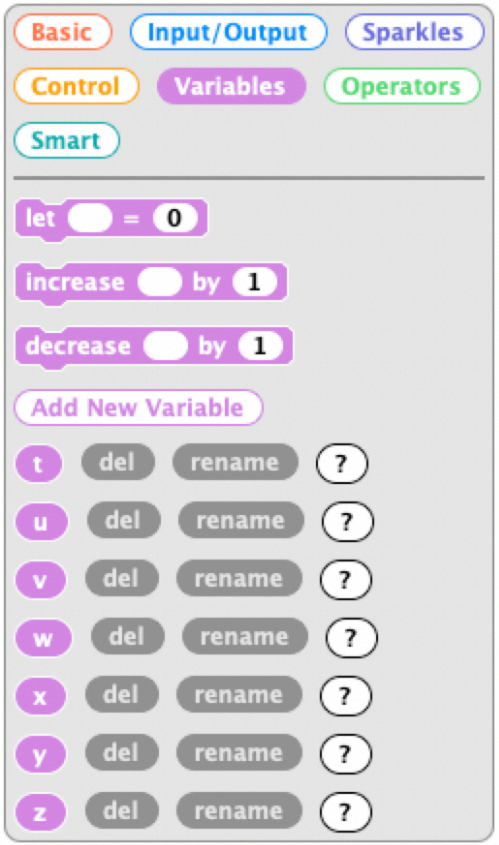
If - then (hacer si): Estructura condicional. El conjunto de bloques se ejecutan si se da la condición que pongamos en el rombo.

If - then - else (hacer si - si no): Estructura condicional. Ejecuta un fragmento de código u otro en función de una condición.

Do until loop (repetir hasta): Ejecuta un conjunto de bloques de programa de forma repetida, hasta que se de la condición que pongamos en el rombo de la estructura.

Do forever loop (repetir por siempre): Ejecuta un conjunto de bloques de programa de forma repetida.

Do times loop (repetir n veces): Ejecuta un fragmento de código un número determinado de veces.



PALETA VARIABLES

Let (asigna): Asigna un valor a unha variable (números enteros no rango de -32768 a 32767)

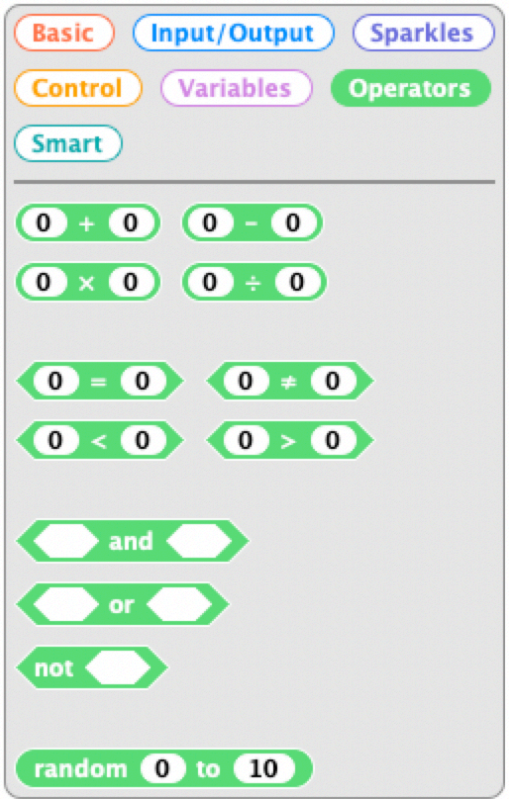
Increase (incrementa): Incrementa o valor dunha variable nunha certa cantidade.

Decrease (decrementa): Decrementa o valor dunha variable nunha certa cantidade.

Xestión de variables:

- **Rename:** renomear variable.
- **Del:** borrar variable.
- **Add New Variable:** engadir nova variable.

O nome dunha variable solo pode conter letras, maiúsculas ou minúsculas. Non pode conter números, espazos en branco, signos de puntuación nin caracteres especiais (p. ej. vocais acentuadas ou letra “ñ”)



PALETA OPERATORS

Suma: Realiza la suma aritmética de dos operandos (el resultado de la operación y los operandos son números enteros dentro del rango -32768 a 32767)

Resta: Realiza la resta aritmética de dos operandos (el resultado de la operación y los operandos son números enteros dentro del rango -32768 a 32767)

Multiplicación: Realiza la multiplicación aritmética de dos operandos (el resultado de la operación y los operandos son números enteros dentro del rango -32768 a 32767)

División: Realiza la división aritmética de dos operandos (el resultado de la operación y los operandos son números enteros dentro del rango -32768 a 32767)

Igualdad: Realiza una comprobación binaria de igualdad. El resultado será **cierto** si los dos operando son iguales y **falso** en caso contrario.

Desigualdad: Realiza una comprobación binaria de desigualdad. El resultado será **cierto** si los dos operando no son iguales y **falso** en caso contrario.

Menor que: Realiza una comprobación binaria *"menor que"*. El resultado será **cierto** si el operando de la izquierda es menor que ele de la derecha y **falso** en caso contrario.

Mayor que: Realiza una comprobación binaria *"mayor que"*. El resultado será **cierto** si el operando de la izquierda es mayor que ele de la derecha y **falso** en caso contrario.

And: Realiza la función lógica AND de las dos condiciones que pongamos en el bloque. El resultado de la función será **cierto**, si lo son las dos condiciones.

Or: Realiza la función lógica OR de las dos condiciones que pongamos en el bloque. El resultado de la función será **cierto**, si lo es cualquiera de las dos condiciones.

Not: Realiza la función lógica NOT de las condición que pongamos en el bloque. El resultado de la función será **cierto**, si la condición es falsa y **falso** si la condición es cierta.

Random: Genera un número aleatorio entero dentro del rango de valores indicados, incluidos estos (rango máximo de -32768 a 32767)